CSC 316 – Data Structures

Programming Project #2 – Family Trees and Relationships

Project Objectives

With this project you will construct a representation of a family tree from listings of its nodes in preorder and postorder, then answer questions about relationships of pairs of individuals (nodes) in the tree. The objectives of the project are to:

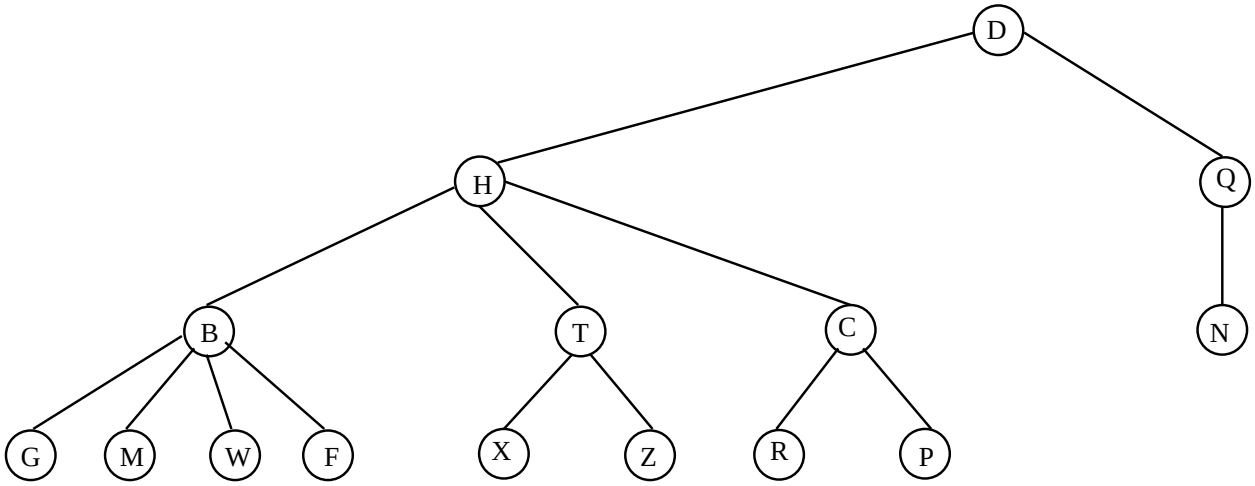
- implement a general tree data structure and the tree ADT operations; and
- gain experience with recursion by writing a recursive procedure to construct the tree from its preorder and postorder traversals.

Input and Output

Typical input for the project will be:

```
< D, H, B, G, M, W, F, T, X, Z, C, R, P, Q, N.
> G, M, W, F, B, X, Z, T, R, P, C, H, N, Q, D.
?  D, H.
?  W, T.
?  N, F.
```

With this input, your program will construct a representation of the following tree:



and print the following output:

```
D is H's parent.
W is T's niece/nephew.
N is F's 1th cousin 1 times removed.
```

The first line of the input is always the preorder traversal, beginning with `<` and ending with a period. The second line is the postorder traversal, beginning with `>`. The following lines contain queries, beginning with `?`.

Constructing a Tree from its Preorder and Postorder Traversals
Your task is to develop and implement a recursive procedure to construct the tree from its preorder and postorder traversals. Write all your code in terms of the tree ADT operations For this project, **you may not use built-in Java classes, you need to implement your own classes**. Choose an appropriate implementation for this application from among those discussed in class. Your program should still work correctly of another implementation is substituted for the one you chose.

Consider an unknown tree with preorder and postorder traversals:

```
H, B, G, M, W, F, T, X, Z, C, R, P
G, M, W, F, B, X, Z, T, R, P, C, H
```

We know that the root of the tree is `H`; it is first in preorder and last in postorder. We also know that the tree has 12 nodes. So we need to figure out how to break the sequence

```
B, G, M, W, F, T, X, Z, C, R, P
G, M, W, F, B, X, Z, T, R, P, C
```

into subtrees. Clearly, `B` is the root of the leftmost subtree of `H`; it appears right after the root `H` in preorder. Also, `B` will be the last node in its subtree's postorder traversal. Therefore, we conclude that the subtree rooted at `B` has five (5) nodes and is described by the traversals

```
B, G, M, W, F
G, M, W, F, B
```

The remaining subtree(s) are contained in the traversals

```
T, X, Z, C, R, P
X, Z, T, R, P, C
```

The next subtree is rooted at `T`, and is described by the traversals

```
T, X, Z
X, Z, T
```

The third and final subtree is described by the traversals

```
C, R, P
R, P, C
```

In effect, we have taken the problem of constructing a tree from its preorder and postorder traversal and subdivided it into a number of similar problems on shorter traversals. When the traversals have length one (1), the answer will be obvious (i.e., the subtree consists of only one node with no children). Therefore, this approach is the basis of a recursive procedure to construct the tree.

Recursive buildTree() Method

You may assume that the input is correct, and that it corresponds to a real tree. Each character represents a different node, so no character will be repeated in a single traversal. Since there are only 256 different characters, this is a bound on the length of the input line.

Read the first two input lines into two global arrays, `pretrav` and `posttrav`. Write the following recursive function:

function `buildTree` (**int** `size`, **int** `prestart`, **int** `poststart`): reference to Root node of subtree
`size` is the number of nodes in the subtree to be built
`prestart` is the place in pretrav where the preorder traversal of this subtree begins
`poststart` is the place in posttrav where the postorder traversal of this subtree begins

How to Determine Relationships

Include a `mark` field in each node. One method to determine the relationship of node A to node B is:

- First, mark all ancestors of A, including A itself.
- Second, search B, B's parent, B's grandparent, etc., until the first marked node is found. This node is the *least common ancestor* of A and B. The relationship of A to B is determined from the lengths of the paths from A and B to the least common ancestor, according to the following table:

A's path length	B's path length	Relation of A to B
0	0	A is B
0	1	A is B's parent
0	2	A is B's grandparent
0	3	A is B's great-grandparent
0	$k > 3$	A is B's $(\text{great})^{k-2}$-grandparent
1	0	A is B's child
2	0	A is B's grandchild
$k \geq 3$	0	A is B's $(\text{great})^{k-2}$-grandchild
1	1	A is B's sibling
1	2	A is B's aunt/uncle
1	$k \geq 2$	A is B's $(\text{great})^{k-2}$-aunt/uncle
2	1	A is B's niece/nephew
$k \geq 2$	1	A is B's $(\text{great})^{k-2}$-niece/nephew
$k \geq 2$	$m \geq 2$	A is B's $(\min(m, k) - 1)$th cousin $\lvert k - m \rvert$ times removed.

- Finally, do not forget to erase the marks of all nodes before you work on another query.

Level Order Traversal of the Family Tree
Once your program has answered the queries included in the input file, it will print the nodes of the family three in *level-order*. As discussed in class, you will need to use a queue to implement the level-order traversal, and you will have to visit nodes of the same depth in left-to-right order.

Deliverables: Source Code To Be Tested By The TA
You will implement a Java program to perform the operations described above. For your implementation, you may assume that the number of distinct nodes contained in the family tree will not exceed 256, and similarly, the number of nodes in an input preorder or postorder sequence will not exceed 256.

The program you submit will prompt the user for the name of the input and output files. Your program will build the family tree from the preorder and postorder traversals, and then will print the answers to the queriers following the first two lines of input. Finally, it will print the nodes of the family tree in level-order.

Submission
Submit all your programs by the due date indicated on the course site.

Grading

buildTree	40 Points
Answers to queries	30 Points
Level-order traversal	30 Points
	100 Points

Important reminder: Homework is an individual, not a group, project. Students may discuss approaches to problems together, but each student should write up and fully understand his/her own solution. Students may be asked to explain solutions orally if necessary.